www.icgst.com

DPCM Soft Modelling in Open-Loop Representations

Fahim Salauddin
Department of Electronic Engineering, La Trobe University
Melbourne, Australia
fahim.salauddin.bd@ieee.org

Abstract

In Differential Pulse Coded Modulation (DPCM) the prediction of the next sample value is formed from the past values. This prediction can be considered as a set of instruction for the quantizer to conduct its next sample value in a particular interval. By using the redundancy in the signal to form a prediction, the region of uncertainty is reduced and the quantization can be performed with a reduced amount of decisions (or bits) for a given quantization level or reduced amount of quantization levels for a given number of decisions (or bits). This paper basically focuses on the software modeling of an open-loop DPCM structure by using MATLAB. During Simulation, comparisons were made in terms of different Signal to Noise ratios by using different number of bits per sample of the uniform quantization process and the prediction orders and their optimal values were determined. By determining these optimal values of number of bits as well as the prediction orders, the implementation of open-loop DPCM in production scenarios such as lossless compression or embedded compression could be made much more efficiently. Standalone graphical user interface (GUI) was developed for this task.[1]

Keywords: *dpcm, open-loop dpcm, quantization, differential pulse coded modulation, prediction error, pulse amplitude modulation, pulse coded modulation.*

Nomenclature

DPCM	Differential Pulse Coded Modulation
m[k]	K^{th} sample of uniform quantization
d[k]	Difference between two corresponding uniform quantization samples
Δv	Quantization interval
L	Quantization level
$\hat{\boldsymbol{m}}[\boldsymbol{k}]$	Estimated value of K^{th} sample

[1] This study has been implemented on MATLAB software at BG320/322 lab. La Trobe University.

1. Introduction

DPCM compression depends on prediction technique, well-conducted prediction techniques leads to good compression rates, in other cases DPCM could mean expansion comparing to regular PCM encoding. There are two structures for DPCM: open-loop and close-loop. This paper is mainly focused on the software implementation of the open-loop DPCM.

DPCM is a computationally efficient means for waveform coding. Various common audio signals exhibit a long-term low-pass spectral characteristic which results in a significant correlation between successive samples of the input signal [1].
In order to address some of the applications of DPCM systems when it comes to audio compression, de-noising, video coding, prediction and communication, these papers could be considered [2, 3, 4, 5, 6].
Mansour et.al in their paper [7] created a simulation of the DPCM and ADM systems in Simulink. But there is still a lack of proper standalone GUI based software solutions to perform DPCM simulations. This research paper deals with this problem by creating a standalone GUI based open-loop DPCM simulation. Also this article deals with the determination of optimal values of bits per sample used in uniform quantization and the prediction orders by comparing SNRs. By doing so the applications mentioned in [2, 3, 4, 5, 6] could be performed more efficiently.

Section 2 deals with the theoretical backgrounds applied to the simulation process. This includes defining the relationship between the successive samples and how transmitting the difference between them helps the overall system to become more efficient, rather than transmitting the signal itself.
In section 3, the actual implementation of the simulation parameters were demonstrated. Various functions that were required to build up the robust GUI system for simulation were discussed in this section. Various combinations of experimental encoder and decoder values were also displayed in tabular format in this section.
Section 4 deals with the simulation results and various plots were constructed on the basis of trade-offs between the bits per sample of the uniform quantization process and the prediction order in this section. Tables displaying the

various signal to noise ratios for trade-offs between the bits per sample of the uniform quantization process and the prediction order were displayed in this section such that their optimal values could be found out.
Finally section 5 deals with the conclusion and possible scope of improvements for the project in future.

2. Theoretical Fundamentals

In analog messages it is possible to make a good guess about the sampled values from the knowledge of the past sample values. That is, the sample values are not independent and generally there is a great amount of redundancy in the Nyquist samples. Proper Exploration of this redundancy leads to encoding a signal with a lesser number of bits. If it is considered as a sample scheme where instead of transmitting the sample values, the system transmits the difference between the successive samples. Thus, if $m[k]$ is the kth sample, instead of transmitting $m[k]$, the system transmits the difference $d[k] = m[k] - m[k-1]$ [8].

At the receiver, knowing $d[k]$ and the previous sample value $m[k-1]$, the system can reconstruct $m[k]$. So it can be see that from the knowledge of difference $d[k]$, the system can reconstruct $m[k]$ iteratively at the receiver. Now the difference between the successive samples is generally much smaller than the sample values. Thus, the peak amplitude m_p of the transmitted values is reduced considerably. Because the quantization interval $\Delta v = mp/L$, for a given L (quantization level) this reduces the quantization interval Δv, thus reducing the quantization noise, which can increase the SNR [9].
The proposed system can improve upon this existing scheme by estimating or predicting the value of the k^{th} sample $m[k]$ from a knowledge of the previous sample values. If this estimate is $\hat{m}[k]$, then it actually transmit the difference (prediction error)$d[k] = m[k] - \hat{m}[k]$. At the receiver also, we determine the estimate $\hat{m}[k]$ from the previous sample values, and then generate $m[k]$ by adding the received $d[k]$ to the estimate $\hat{m}[k]$. So it is basically able to reconstruct the samples at the receiver iteratively. If everything has been done properly then the predicted or estimated value $\hat{m}[k]$ will be close to $m[k]$ and their difference (prediction error) $d[k]$ will be even smaller than the difference between the successive samples.

3. Implementation

In DPCM coding a difference is based on the fact that most source signals shows significant correlation between successive samples so encoding uses redundancy in sample values which implies lower bit rate. Realization of basic concept (described above) is based on technique in which we have to predict current sample value based upon previous samples (or sample) and the system has to encode the difference between actual value of sample and predicted value (difference between samples can be interpreted as prediction error).

The open loop DPCM is comprised of mainly two main parts, the Encoder and the decoder. The encoding functionalities are implemented inside the encoder in the transmitting side while the decoding activities are performed by the decoder in the receiver side. Basically the encoder instead of encoding the input signal directly and sending it via the channel to the receiver, it encodes the difference between the main input signal sample and it's linear prediction sample and sends it to the receiver for decoding. This is a much more efficient way of sending information from the transmitter and the receiver as it prevents a lot of redundant calculations. Due to this less number of bits will be needed to send the information via the channel so there will be less chance of bit error probability too.

The functionalities inside the encoder can be explained with the help of following equations that occurs inside the encoder.

$$x_p = \sum_{k=1}^{M} \alpha_k x[n-k] \quad (1)$$

$$e[n] = x[n] - x_p[n] \quad (2)$$

$$e_q[n] = Q(e[n]) \quad (3)$$

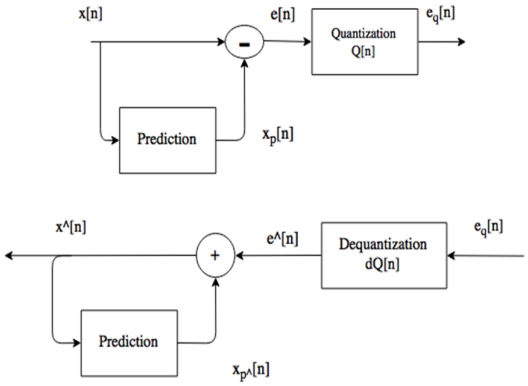

Figure 1: Encoder (top) and Decoder (bottom) block diagrams

Table1: Encoder Experimental values

n	1	2	3	4
x[n]	10	20	20	40
x_p[n]	0	9	18	18
e[n]	10	11	2	22
e_q[n]	2	2	0	5

For the experimental implementation of the encoder, the system basically takes an input signal $x[n] = \{10,20,20,40\}$ The quantizer is designed as $e_q = floor(\frac{e[n]}{4})$. So it can be seen that after the input signal

traverses the entire block diagram of the encoder we are left with the values $e_q = \{2,2,0,5\}$. The encoder basically transmit e_q than the actual signal $x[n]$.
The system can find the experimental values of the functioning of the decoder the same way as the encoder. The de-quantization block is designed as $dQ = 4 * e_q[n]$. Following is the table showing the experimental values of the decoder block.

Table 2: Decoder Experimental Values

n	1	2	3	4
$e_q[n]$	2	2	0	5
$\hat{e}[n]$	8	8	0	20
$\widehat{x_p}[n]$	0	7.2	13.68	12.312
$\hat{x}[n]$	8	15.2	13.68	32.312

The proposed system can also find the error due to excessive quantization from the above experimental values which is given by equation (4). For this the above case it is found to be $E[n] = \{2,5,6,7\}$.

$$E[n] = x[n] - \hat{x}[n] \qquad (4)$$

Separate MATLAB functions entitled;

- *myQuantization,*
- *myDeQuantization,*
- *myPrediction*
- *myPredictionReconstruction*

were built to perform the functions of the block diagrams of the encoder and the decoder. A graphical user interface was also build that gave the user to choose between the three sound samples of handel, laughter and train in pop-up menu.

At first the system loaded the sound file *laughter* with sampling frequency of 8192 Hz in the main mat file. The users were asked in the GUI to enter the Quantization level (K) as well as the Prediction Order (N). As advised in the handouts we performed the encoding and the decoding process by using the Prediction Order of 3 (N =3) and Quantization Level of 4 (K = 4). Following graphs were obtained:

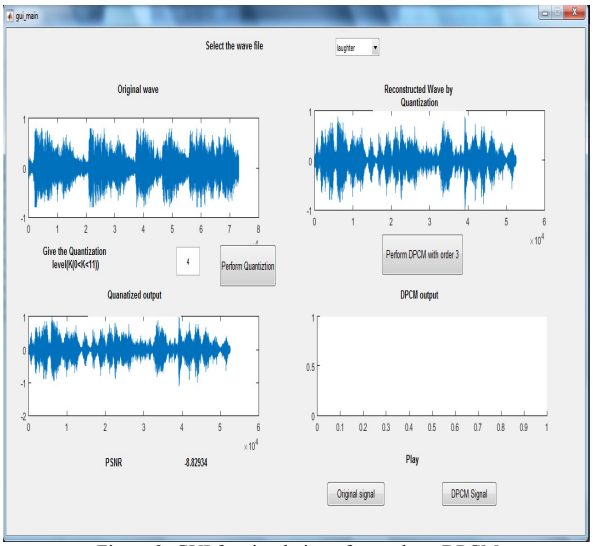

Figure 2: GUI for simulation of open-loop DPCM

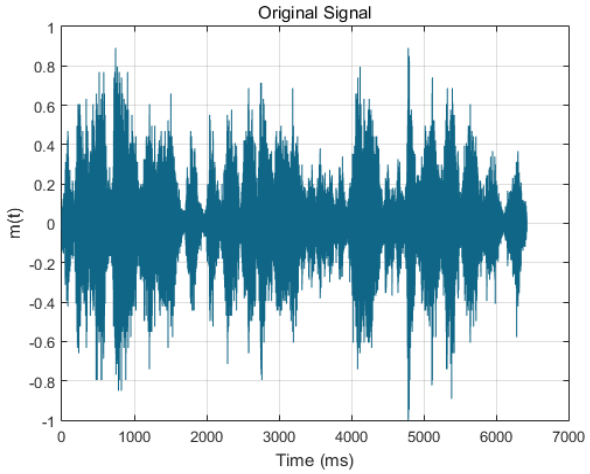

Figure 3: Original Signal (laughter)

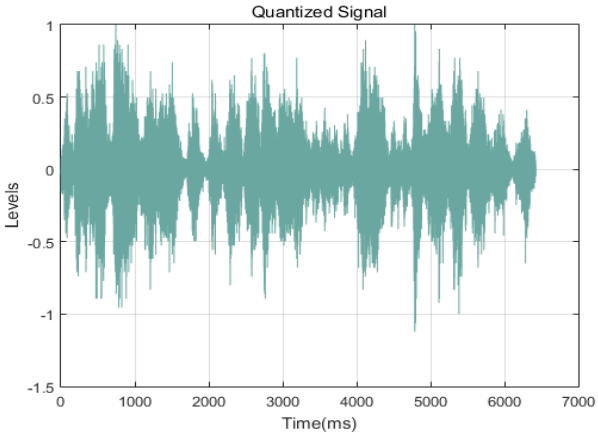

Figure 4: Quantized Signal (laughter)

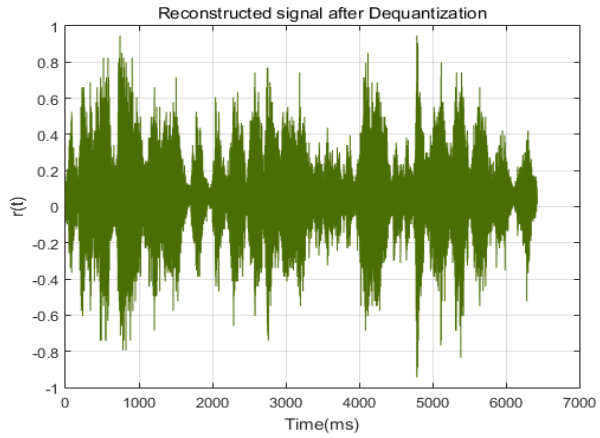

Figure 5: Signal after De-quantization (laughter)

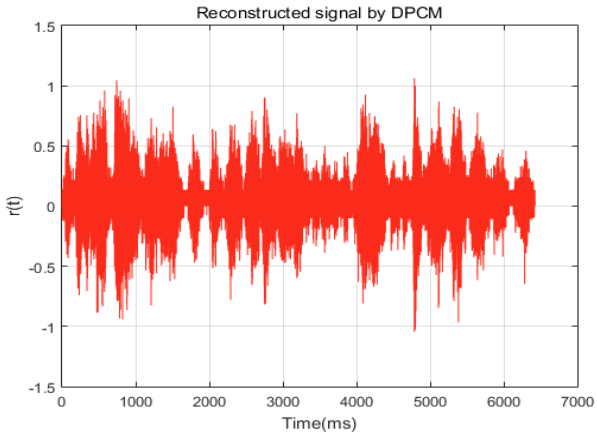

Figure 6: Signal after Linear prediction by order 3 (laughter)

In a similar way the other two signals *handel* and *train* could also be loaded in the main matlab file and it would ultimately give us the SNR that allowed us to estimate which prediction order or bits per sample are appropriate to re-generate the signal as close to the original signal as possible.

SNR was calculated by the given formula :

$$SNR = 10log_{10}\frac{\sum y[n]^2}{\sum(y[n]-r[n])^2} \qquad (5)$$

4. Experimental Results

In the experimental section I basically wanted to choose a suitable prediction order and the number of bits/sample to make a good compromise between the compression and the sound quality. We did that by comparing the SNR values of obtained from the following equation.

$$SNR = 10log_{10}\frac{\sum y[n]^2}{\sum(y[n]-r[n])^2} \qquad (6)$$

Which is just a modified version of the following equation:

$$SNR = -10log_{10}\frac{\sum(r[n]-y[n])^2}{\sum(y[n]^2} \qquad (7)$$

I started the procedure by keeping $K = 1$ and calculated the values of both *SNR_r* and *SNR_e* for $1 \leq N \leq 5$.

Table 3: SNR Comparison for k=1 and 1≤N≤5

K	N	SNR_r
1	1	-22.1855
1	2	-16.1649
1	3	-10.2901
1	4	-6.1562
1	5	-4.6608
1	6	-4.3322

I observed from the table that as I increased the prediction order N, the *SNR_r* increased, which is desirable as the system was able to produce a sound quality much similar to the original signal. The system was thought to have a better Compression i.e. a better prediction then. As I saw that the system was getting a better sound quality as well as a better compression for both the increasing K and N I constructed the following table to get the optimal SNR_r and SNR_e values at the range of $1 \leq K \leq 11$ and $1 \leq N \leq 10$.

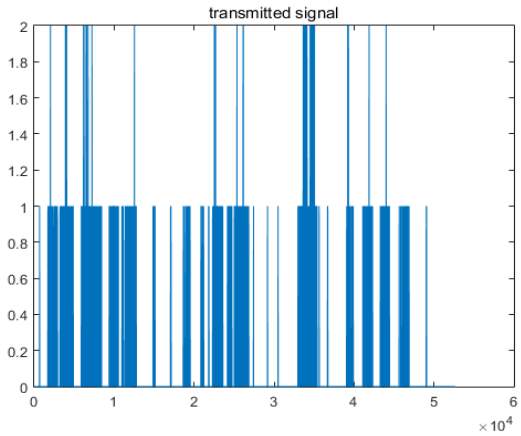

Figure 7: Transmitted Signal for K = 4 N = 4

Table 4: SNR Comparison for 1≤K≤11 and 1≤N≤11 (laughter)

K	N	SNR_r
1	1	-22.1855
2	2	-11.6936
3	3	-6.6188
4	4	-2.9897
5	5	-2.2874
6	6	-1.9291
7	7	-1.7244
8	8	-1.6341
9	9	-1.6012
10	10	-1.4655
11	11	-1.4442

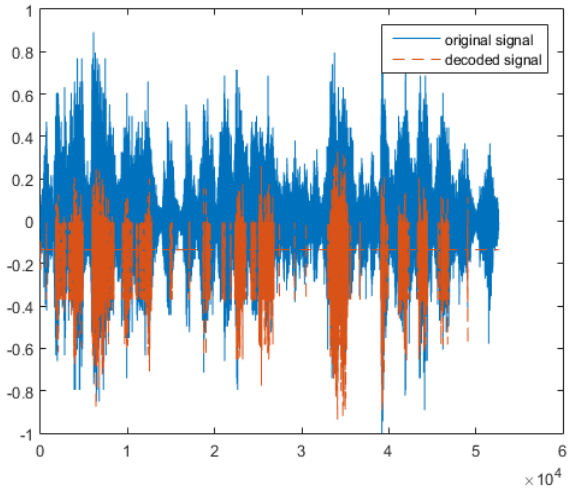

Figure 8: Original Vs. Decoded signal for K = 4 N = 4

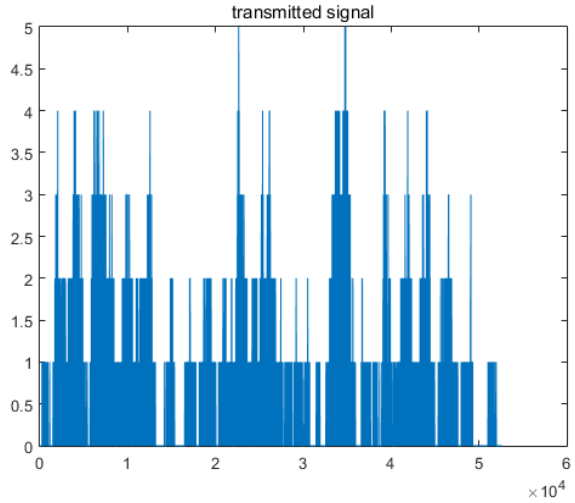

Figure 9: Transmitted Signal for K = 5 N = 5

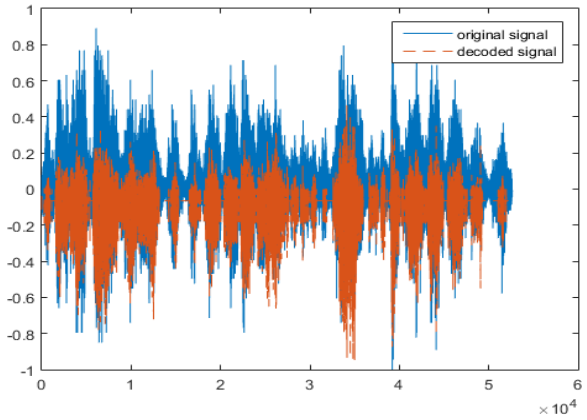

Figure 10: Original Vs. Decoded signal for K = 5 N = 5

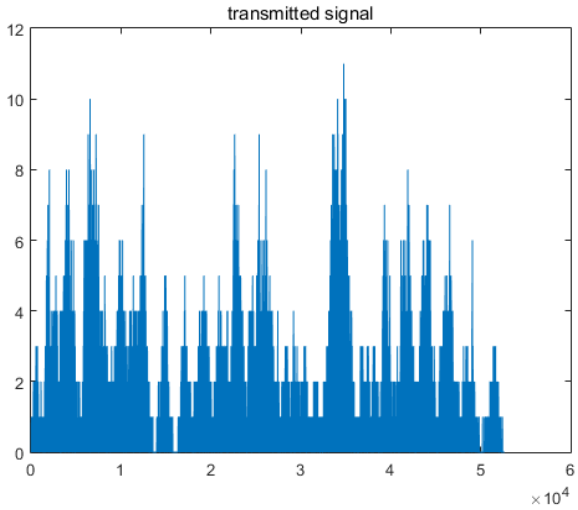

Figure 11: Transmitted Signal for K = 6 N = 6

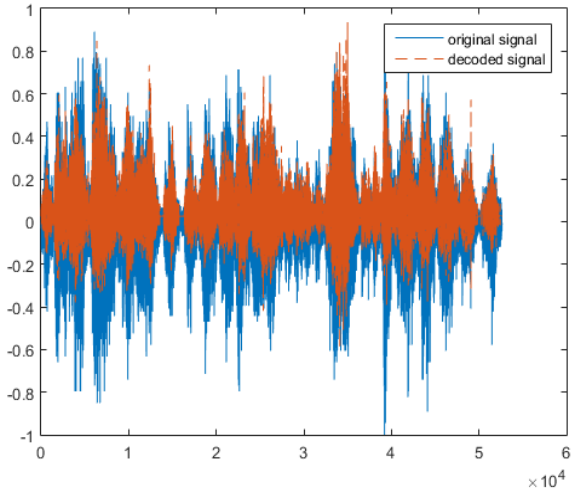

Figure 12: Original Vs. Decoded signal for K = 6 N = 6

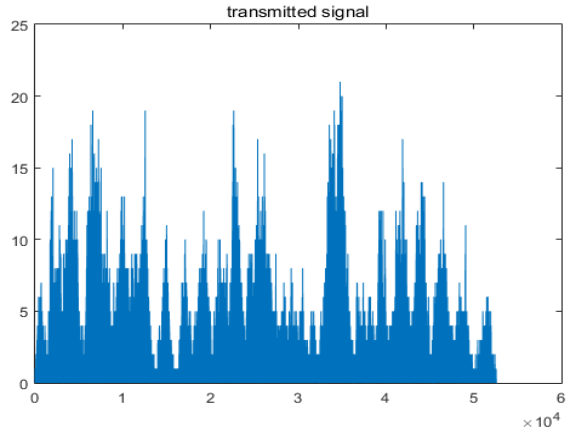

Figure 13: Transmitted Signal for K = 7 N = 7

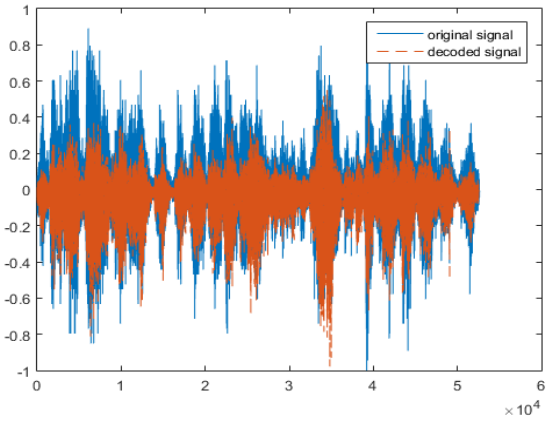

Figure 14: Original Vs. Decoded signal for K = 7 N = 7

Table 5: SNR Comparison for 1≤K≤10 and 1≤N≤10 (Handel)

K	N	SNR_r
1	1	-24.4608
2	2	-14.7124
3	3	-8.6330
4	4	-4.6088
5	5	-4.3354
6	6	-3.4003
7	7	-2.9174
8	8	-2.3727
9	9	-2.1369
10	10	-2.0144

Table 6: SNR Comparison for 1≤K≤10 and 1≤N≤10 (Train)

K	N	SNR_r
1	1	-19.1465
2	2	-8.2138
3	3	-3.2407
4	4	-1.3343
5	5	-1.3125
6	6	-1.4291
7	7	-1.5104
8	8	-1.0929
9	9	-1.1158
10	10	-0.7450

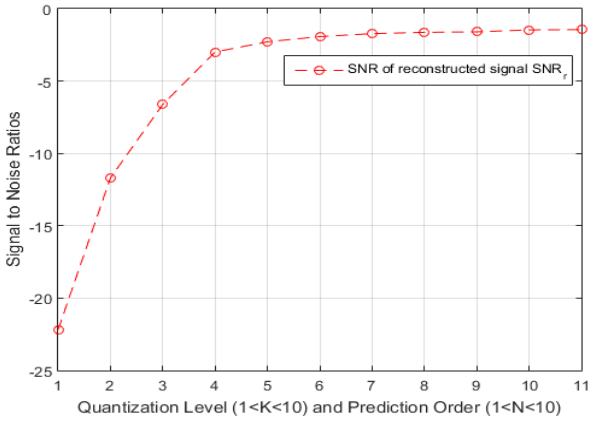

Figure 15: Signal to Noise ratios for both r and e against (1<k<11) and (1<N<11) for laughter signal

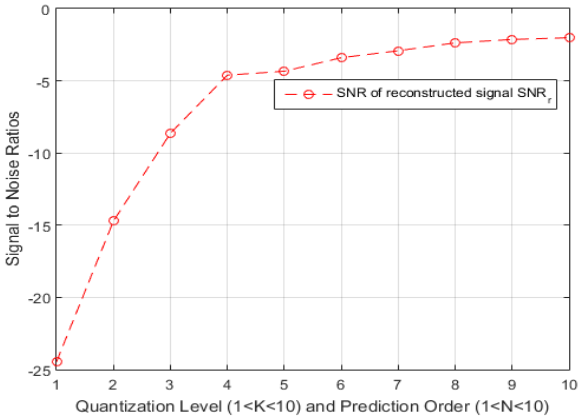

Figure 16: Signal to Noise ratios for both r and e against (1<k<10) and (1<N<10) for handel signal

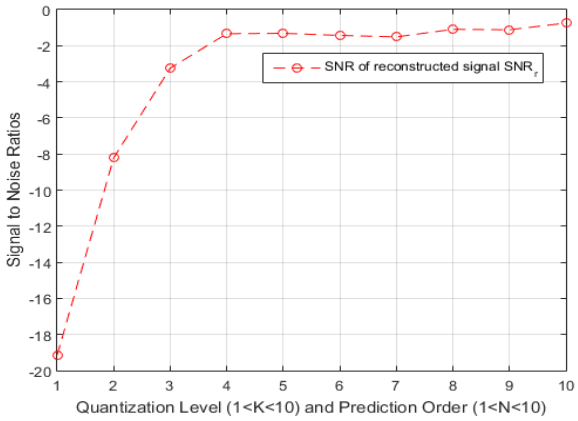

Figure 17: Signal to Noise ratios for both r and e against (1<k<10) and (1<N<10) for train signal

From the graph above it can be said that SNR_r is maximum for both K = 10 and as well as N = 10. Whereas the prediction error is lowest for lowest for K = 10 and N = 10. So it can be said that when the signal is quantized using increasing number of bits per sample as well as a higher prediction order it can have a proper audible reconstruction of the signal as well as a better compression due to decreasing e[n].

It can also be seen that both the SNRs approximately intersect each other in a point when both *(K, N) = (4, 4)*. So the optimum point is seen somewhere around *(K, N) = (4, 4)* for both the *laughter* and the *handel* signals. But for *train* signal the optimum point will be at *(K, N) = (5, 5)*.

Table 7: Optimum SNR for trade-off between K(2-11) and N(1-6) for TRAIN Signal

|N| K	0	1	2	3	4	5	6
2	-19.59	-13.71	-8.21	-3.82	-1.84	-1.61	-1.61
3	-18.56	-12.72	-7.35	-3.24	-1.63	-1.44	-1.41
4	-17.40	-11.60	-6.41	-2.66	-1.33	-1.32	-1.32
5	-17.41	-11.62	-6.42	-2.67	-2.67	-1.31	-1.32
6	-16.10	-10.61	-5.42	-2.10	-1.20	-1.44	-1.43
7	-13.42	-7.96	-3.64	-1.24	-0.70	-1.38	-1.49
8	-9.51	-4.75	-1.76	-0.51	-0.25	-1.02	-1.01
9	-8.16	-3.78	-1.30	-0.37	-0.12	-0.81	-1.02
10	-5.80	-2.31	-0.70	-0.19	-0.04	-0.75	-0.68
11	-11.56	-6.37	-2.64	-0.83	-0.22	-0.65	-0.78

Table 8: Optimum SNR for trade-off between K(2-11) and N(7-10) for Train Signal

|N| K	7	8	9	10
2	-1.62	-1.63	-1.63	-1.63
3	-1.42	-1.43	-1.43	-1.42
4	-1.33	-1.33	-1.33	-1.33
5	-1.32	-1.32	-1.33	-1.33
6	-1.43	-1.43	-1.42	-1.42
7	-1.51	-1.59	-1.53	-1.54
8	-1.05	-1.09	-1.09	-1.09
9	-1.03	-1.10	-1.11	-1.12
10	-0.73	-0.75	-0.75	-0.74
11	-0.79	-0.72	-0.76	-0.77

So it can be seen from the table above that the optimum at Prediction Order, N = 4 and Uniform quantization of Bits per sample, K = 10 we got the highest SNR of -0.04 for *train* signal. Similarly the optimum SNR for trade-off between K and N for *laughter* as well as *handel* signals could be found. As these optimal values of K and N are determined for the three signals *train, laughter* and *handel*. Similarly this custom built GUI and MATLAB functions could be applied to any functions with after loading them as raw data inside MATLAB. So as discussed earlier in the introduction section the applications mentioned in the papers [2, 3, 4, 5, 6] could use the simulation process of this paper and increase the efficiency of their applications by using the optimal values of the bits per sample of the uniform quantization process in their applications.

5. Conclusion

This system successfully recreated both the encoder and the decoder of open-loop DPCM by applying some custom built MATLAB functions. It was basically used to find the trade-offs between the bits per sample of the uniform quantization process and the prediction order by comparing their SNRs. The results that were obtained could be used in variety of applications such as compression, de-noising, video coding, prediction, interpolation, and communications. The proposed technique of modeling and simulation of this paper is restricted to the application to DPCM simulation only. For an improvement to this paper, the proposed modeling and simulation technique could be applied to wide variety of digital filters like Kalman filter, wiener filter, LMS, RLS, AR, ARX, ARMA, ARMAX, IIR, FIR, FFT, IFT, DCT, DST etc.

6. References

[1] N. S. Jayant, "Digital coding of speech waveforms: PCM, DPCM, and DM quantizers," Proc. IEEE, vol. 62 (May, 1974), pp. 611-632.

[2] N. S. Jayant and P. Noll, Digital Coding of Waveforms, Englewood Cliffs, NJ: Prentice-Hall, 1984.

[3] B. S. Atal and M. R. Schroeder, "Predictive coding of speech signals and subjective error criteria," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-27 (June 1979), pp. 247-254.

[4] J. C. Asmuth and J. D. Gibson, "Sequential noise spectral shaping in DPCM," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, no. 2 (1984), pp. 228-235.

[5] R. C. Maher, "Computationally efficient compression of audio signals by means of RIQ-DPCM," Applications of Signal Processing to Audio and Acoustics, 1993. Final Program and Paper Summaries., 1993 IEEE Workshop on, New Paltz, NY, 1993, pp. 35-38.

[6] R. Achkar, G. A. Haidar and C. Mansour, "Real-time application of DPCM and ADM systems," Communication Systems, Networks & Digital Signal Processing (CSNDSP), 2012 8th International Symposium on, Poznan, 2012, pp. 1-6.

[7] C. Mansour, R. Achkar and G. A. Haidar, "Simulation of DPCM and ADM Systems," Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on, Cambridge, 2012, pp. 416-421.

[8] B. P. Lathi. 1998. Modern Digital and Analog Communication Systems 3e Osece (3rd ed.). Oxford University Press.

[9] Bernard Sklar. 1988. *Digital Communications: Fundamentals and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[10] S. Hoque, F. Salauddin and A. Rahman, "Neighbour cell list optimization based on cooperative q-learning and reinforced back-propagation technique," Radio Science Meeting (Joint with AP-S Symposium), 2015 USNC-URSI, Vancouver, BC, Canada, 2015, pp. 215-215.

[11] S. Rahman, A. Ahmed, F. Salauddin and A. Rahman, "HetNet performance analysis with asynchronous ABSF configuration employing horizontal sector offset scheme," Radio Science Meeting (Joint with AP-S Symposium), 2015 USNC-URSI, Vancouver, BC, Canada, 2015, pp. 216-216.

[12] N. Hossain, M. T. Kabir, T. R. Rahman, M. S. Hossen and F. Salauddin, "A real-time surveillance mini-rover based on OpenCV-Python-JAVA using Raspberry Pi 2," 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, 2015, pp. 476-481.

[13] F. Salauddin and T. R. Rahman "A Fuzzy based low-cost monitoring module built with raspberry pi - python - java architecture", International Conference on Smart Sensors and Application (ICSSA

Biography

Fahim Salauddin completed his BEng. Degree in Electronics and Telecommunications Engineering from North South University, Dhaka, Bangladesh. Has multiple IEEE publications in Heterogeneous Networks and embedded systems. Currently pursuing his Master in Telecommunications and Network Engineering from La Trobe University, Melbourne, Australia. His research interest lies in Wireless Communications, Software Defined Networking, Internet of Things, Cloud Computing and embedded systems.